periodica polytechnica

Electrical Engineering and Computer Science
56/2 (2012) 43–50
doi: 10.3311/PPee.7162
http://periodicapolytechnica.org/ee

RESEARCH ARTICLE

Received 2012-03-12, accepted 2012-07-27

A Model Predictive Navigation Approach Considering Mobile Robot Shape and Dynamics

Domokos Kiss / *Gábor* Tevesz

Abstract

Most mobile robot navigation approaches assume the robot being point-like or consider only its bounding circle while looking for a collision-free path to a given goal position. A well-known method called the Dynamic Window Approach (DWA) introduced an interesting idea for solving the navigation problem by local optimization in the control space of the robot. Some extensions of the original DWA method can also be found in the literature, which enable its applicability to holonomic and non-holonomic robots and ensure a global and safe solution to the navigation problem. The method described in this paper has also been motivated by the basic idea of dynamic window and contributes to the previous variants by taking the robot shape into consideration as well. A navigation function based model predictive control scheme is utilized to choose the appropriate control for a safe and successful navigation.

Keywords

Dynamic Window Approach · Model Predictive Control · Configuration Space · Holonomic Robot

Acknowledgement

The work in the paper has been developed in the framework of the project "Talent care and cultivation in the scientific workshops of BME". This project is supported by the grant TÁMOP - 4.2.2.B-10/1–2010-0009.

Domokos Kiss

Department of Automation and Applied Informatics, BME, H-1117 Budapest, Magyar Tudósok körútja 2., Hungary
e-mail: domokos.kiss@aut.bme.hu

Gábor Tevesz

Department of Automation and Applied Informatics, BME, H-1117 Budapest, Magyar Tudósok körútja 2., Hungary
e-mail: tevesz@aut.bme.hu

1 Introduction

Mobile robot navigation in real-world scenarios has gained increasing interest and importance in the last years as mobile robot applications have spread to many areas of research, industry and even our everyday life. The navigation problem consists of more subproblems that need to be solved for reaching a predefined goal safely and successfully. At first, a mobile robot that operates in a real environment has to be able to recognize obstacles around it and build a map of the environment. It is also necessary to handle moving obstacles and changes in the environment. The second task is to determine its own position inside the navigation scene. Finally, it has to plan and accomplish a motion sequence that ends at a given goal while avoiding collision with obstacles. There are different approaches for solving these subproblems. For example, mapping and localization can be done simultaneously as shown in [11, 12]. The solution of the simultaneous localization and mapping problem (SLAM) is subject to active research but is out of the scope of this paper. In other words, we assume that the environment map and robot position is known at any time during navigation.

This paper focuses on the motion planning task of a mobile robot moving in the plane, in the case of such obstacle distributions where narrow crossings are unavoidable. The majority of planning approaches consider the robot as a single point or take the dimensions of it into account by replacing the robot with its bounding circle. This assumption works if there are wide free areas between robot and its target, but fails if the robot itself is not circle-shaped and the bounding circle is too large to pass narrowings.

An elegant real-time planning strategy, the *dynamic window approach* (DWA) is proposed in [5]. Some extensions of this approach have also been published in the past years but in most cases the vehicle shape is not taken into account. The contribution of this paper is a dynamic window-based navigation method that utilizes a global navigation function to ensure reaching the goal and takes the shape of the robot into account as well.

The paper is organized as follows. In Section 2, we give a short survey of dynamic window-based navigation approaches. In Section 3, it is shown how the shape of the robot is taken

into account in the proposed approach. Section 4 is about the navigation function that plays an important role in the proposed method. In Section 5, we describe the model predictive algorithm and show simulated experimental results. Section 6 summarizes the paper and gives directions of future work.

2 Related Work and Background

In the past decades, several mobile robot navigation techniques were proposed. Early approaches based on artificial potential fields have drawn great attention among researchers. The idea of using virtual forces to act on the robot were utilized in [2–4] among others. The simplicity and elegance of potential function approaches made them very popular. However, after a few years of extensive research and experimental work, it was shown that these techniques have some inherent limitations [15] which encumber their application to real robots. One of the limitations is that they generate motion commands for the robot in two separate stages. First, the desired direction is determined, and the steering commands that result a motion in this direction are generated in a second step. The dynamic constraints of the robot are not taken into account or in other words they presume that arbitrary forces can be asserted on the robot to achieve a motion in the chosen direction.

In order to solve this problem, methods like the dynamic window approach [5] and curvature velocity method [16] were proposed. These differ from former approaches in that they assume a velocity motion model for robots, i.e. velocities are considered as actuating variables. They deal with robots having nonholonolmic kinematic constraints, whose trajectories can be approximated by a sequence of circular arcs. A circular path segment can be characterized by a velocity pair (v, ω) which consists of the translational velocity v and the angular velocity ω of the robot. These approaches take the dynamics of the robot into account by reducing the search space to velocities reachable in a short time interval. This subset of the velocity space is called the *dynamic window*. In addition, only those velocities are considered that are safe with respect to obstacles (admissible velocities). To choose from the set of admissible velocities an objective function is evaluated and maximized. This objective function in [5] is a weighted sum of three terms: heading(v, ω), which is a measure of going into the direction of the goal, dist(v, ω) being the smallest distance to the next obstacle along the circular path segment belonging to (v, ω) and velocity(v, ω), simply the projection of (v, ω) on the translational velocity v (to favor high motion speeds).

Experimental results presented in [5] show that the dynamic window approach to collision avoidance yields a fast and safe robot motion. However, since the DWA and the above mentioned other methods are based on local decisions without taking connectivity information of free space into account, the robot can get trapped in local minima situations (i.e. it can stop far from the goal point) or enter a limit-cycle that prevents reaching the goal. Another problem of the DWA is that different situations require different weighting of the objective function terms to ensure successful motion but there is no algorithm for choosing the weights.

A modified approach, called the Global DWA [7] extends the original method to the case of holonomic robots and addresses the problem of local minima by taking free space connectivity information into account. This is obtained by introducing a navigation function (NF), which is a local minima-free function defined on the discretized configuration space, having a unique minimum at the goal. New terms are added to the objective function to favor NF descent along the robot path.

The problem of local minima is eliminated in many cases through the global distance information represented by the NF terms, but not at all times. It is shown in [10] that limit-cycles can evolve if the velocity term outweighs the NF terms. They reformulate the dynamic window approach as a model predictive control (MPC) problem (also referred to as receding horizon control, RHC). They assume a holonomic robot model that can be considered a double integrator in the plane $\ddot{r} = u$, where $r \in \mathbb{R}^2$ is the position of the robot. Thus, the control u is the acceleration which is in contrast with [5] where the velocity is the control signal. The state vector of the holonomic robot is chosen as $x = (r, \dot{r}) = (r_x, r_y, \dot{r}_x, \dot{r}_y)$. The acceleration u and the velocity $\dot{r}$ are bounded. The objective function in [10] has a different form as compared to [5] and [7]. It was shown that for a given set of u the system is stable in the Lyapunov sense (but not asymptotically). However, this property is not enough to ensure convergence, since the case of $\|\dot{r}\| = 0$ far from the goal cannot be excluded. This is prevented by a timeout condition in their proposed algorithm.

Another improvement of the original DWA is presented in [13], called I-DWA. This variant adds convergence improvements to the original method by pre-calculating an ideal control action that would make the robot converge to the goal if no obstacles were present. The objective function is similar to the one in [5] and favors control actions close to the ideal ones. Because no global information about obstacles is taken into account, the convergence is not actually ensured.

The above mentioned approaches have the same property of assuming a point-like or circle-shaped robot. A circle-shaped robot can also be reduced to a point if the obstacles are dilated by the robot radius. This is on one hand a very effective and simple assumption if the robot actually has a circular shape or the bounding circle can be used for representing the robot in the collision detection phase of the algorithm. On the other hand, if the shape considerably differs from a circle and there are narrow passages between obstacles, this assumption fails and the algorithms report that no collision-free path exists.

To overcome this limitation, some approaches were also proposed that take the vehicle shape into account. In [6] robot shape is handled by using precalculated lookup tables that contain local data about collision risk depending on the current velocity and obstacle configuration. This approach is memory intensive

but a real-time local obstacle avoidance can be obtained. The approach in [8] solves the task analytically for polygonal robots without the use of lookup tables. These two approaches assume that obstacles are represented by points (e.g. obtained by laser range sensors). This representation is efficient only in the case of local navigation taking only the neighborhood of the robot into consideration, because in a global case too many obstacle points would have to be stored and handled. The method presented in [14] uses the same obstacle model. Unlike others, it delivers an elegant and analytical solution by "wrapping" former existing obstacle avoidance approaches (e.g. potential field methods) in a framework that allows consideration of robot shape and kinematic and dynamic constraints, while it still has the limitation of locality. Since only local methods can be "wrapped", their inherent convergence problems are not solved.

The authors of the present paper have also proposed a DWA-based approach for mobile robot navigation, the Global Dynamic Window Approach using Receding Horizon Control (GDWA/RHC) [17, 18]. It works in velocity space and assumes a non-holonomic and circular robot model, similar to [5] and [13]. Global information is taken into account by a local-minima-free navigation function which serves as a basis for optimization. The appropriate control is chosen from the actual dynamic window by a model predictive method. The objective function has no weighted terms and the control law looks like as follows:

$$\mathbf{u}(\cdot) = \arg\min_{\mathbf{u}(\cdot)} \mathrm{NF}\left(r_x(t+T), r_y(t+T)\right) \tag{1}$$

where $\mathrm{NF} : \mathbb{R}^2 \to \mathbb{R}$ is the navigation function defined on the configuration space of the robot, $r_x(t+T)$ and $r_y(t+T)$ are the predicted robot position coordinates at the end of a time horizon, which can be derived from the motion equation of the robot.

3 Consideration of Robot Shape

In the remaining part of the paper we describe a model predictive navigation strategy that also makes use of a local-minima-free navigation function to ensure reaching the target successfully. In addition to the above mentioned approaches it takes the shape of the robot into account as well. The obstacle representation is not restricted to points, but a polygonal obstacle model is used instead.

3.1 Robot Model

The shape of the robot is assumed to be polygonal. If considering only planar motion, the position and orientation of the robot in the global coordinate system (also referred to as configuration **q**) can be described by three independent position and orientation coordinates $\mathbf{q} = [x, y, \varphi]^T$. We further assume that the robot is holonomic which means that it has three degrees of freedom as well, i.e. it is fully actuated. The robot model is illustrated in Figure 1. The orientation φ is the angle between the x-axes of the global and the robot coordinate frame. Since the robot is holonomic, the directions of the translational velocity **v** and acceleration **a** are independent of the robot orientation. The velocity direction is denoted by ϑ, the direction of the acceleration **a** relative to **v** is denoted by α.

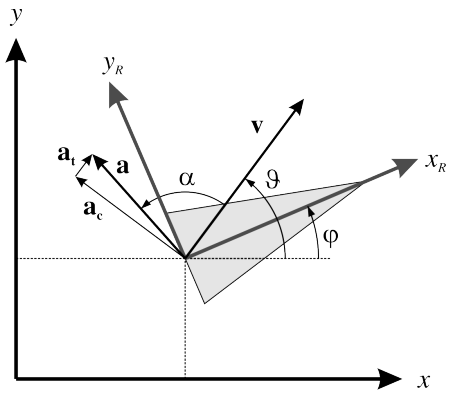

Fig. 1. Robot model

The control signal $\mathbf{u} = [a_x, a_y, \beta]^T$ is the vector of translational and angular accelerations. Under this assumption the robot can be considered as a double integrator in the configuration space and its motion equation has the following simple form:

$$\ddot{\mathbf{q}} = \mathbf{u} \tag{2}$$

Our goal is to control the robot to a given goal configuration $\mathbf{q}_g$ and it has to stop there, i.e. $\dot{\mathbf{q}} = 0$. For that reason we choose the state vector of the system to $\mathbf{x} = [\mathbf{q}, \dot{\mathbf{q}}]^T = [x, y, \varphi, v_x, v_y, \omega]^T$ and the state equation becomes

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & \mathbf{I} \\ 0 & 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ \mathbf{I} \end{bmatrix} \mathbf{u}. \tag{3}$$

It can be written in scalar form as well:

$$\begin{aligned} \dot{x} &= v_x = |\mathbf{v}| \cdot \cos\vartheta \\ \dot{y} &= v_y = |\mathbf{v}| \cdot \sin\vartheta \\ \dot{\varphi} &= \omega \\ \dot{v}_x &= a_x = |\mathbf{a}| \cdot \cos(\alpha + \vartheta) \\ \dot{v}_y &= a_y = |\mathbf{a}| \cdot \sin(\alpha + \vartheta) \\ \dot{\omega} &= \beta \end{aligned} \tag{4}$$

where v_x, v_y, a_x, a_y are the Cartesian coordinates of translational velocity and acceleration in the global frame, respectively. ω stands for angular velocity, β for angular acceleration of the robot.

The dynamic properties of the robot are taken into account by constraints on velocity and acceleration magnitudes:

$$\begin{aligned} |\mathbf{v}| &\le v_{\max} \\ |\omega| &\le \omega_{\max} \\ |\mathbf{a}| &\le a_{\max} \\ |\beta| &\le \beta_{\max} \end{aligned} \tag{5}$$

For simplicity, the translational velocity and acceleration magnitudes $|\mathbf{v}|$ and $|\mathbf{a}|$ will be written as v and a in the sequel.

Since we have magnitude constraints on translational velocity and acceleration, it makes sense to use polar coordinates for these instead of Cartesian ones. Let us redefine the state vector as $\mathbf{x} = [x, y, \varphi, v, \vartheta, \omega]^T$, the control vector as $\mathbf{u} = [a, \alpha, \beta]^T$ and perform the following change of coordinates:

$$v = \sqrt{v_x^2 + v_y^2} \tag{6}$$

$$\vartheta = \arctan(v_y / v_x) \tag{7}$$

The time derivative of (6) is obtained as follows:

$$\begin{aligned}
\dot{v} &= \frac{1}{2}\left(v_x^2 + v_y^2\right)^{-\frac{1}{2}} \cdot \left(2v_x\dot{v}_x + 2v_y\dot{v}_y\right) = \\
&= \frac{v_x a\cos(\alpha+\vartheta) + v_y a\sin(\alpha+\vartheta)}{\sqrt{v_x^2+v_y^2}} = \\
&= \frac{v_x a\cos\alpha\cos\vartheta}{\sqrt{v_x^2+v_y^2}} - \frac{v_x a\sin\alpha\sin\vartheta}{\sqrt{v_x^2+v_y^2}} \\
&\quad + \frac{v_y a\sin\alpha\cos\vartheta}{\sqrt{v_x^2+v_y^2}} + \frac{v_y a\cos\alpha\sin\vartheta}{\sqrt{v_x^2+v_y^2}} = \\
&= \frac{a\cos\alpha\cdot v\cos^2\vartheta}{v} - \frac{v\cos\vartheta\cdot a\sin\alpha\sin\vartheta}{v} \\
&\quad + \frac{v\sin\vartheta\cdot a\sin\alpha\cos\vartheta}{v} + \frac{a\cos\alpha\cdot v\sin^2\vartheta}{v} = \\
&= a\cos\alpha\left(\cos^2\vartheta + \sin^2\vartheta\right) = \\
&= a\cdot\cos\alpha
\end{aligned} \tag{8}$$

Let us do the same for (7):

$$\begin{aligned}
\dot{\vartheta} &= -\frac{v_y}{v_x^2+v_y^2}\cdot\dot{v}_x + \frac{v_x}{v_x^2+v_y^2}\cdot\dot{v}_y \\
&= -\frac{v_y}{v^2}\cdot a\cos(\alpha+\vartheta) + \frac{v_x}{v^2}\cdot a\sin(\alpha+\vartheta) \\
&= -\frac{v_y a(\cos\alpha\cos\vartheta - \sin\alpha\sin\vartheta)}{v^2} \\
&\quad + \frac{v_x a(\sin\alpha\cos\vartheta + \cos\alpha\sin\vartheta)}{v^2} \\
&= -\frac{v\sin\vartheta\cdot a\cos\alpha\cos\vartheta}{v^2} + \frac{a\sin\alpha\cdot v\sin^2\vartheta}{v^2} \\
&\quad + \frac{a\sin\alpha\cdot v\cos^2\vartheta}{v^2} + \frac{v\cos\vartheta\cdot a\cos\alpha\sin\vartheta}{v^2} \\
&= \frac{1}{v}\cdot a\cdot\sin\alpha\cdot\left(\sin^2\vartheta + \cos^2\vartheta\right) \\
&= \frac{1}{v}\cdot a\cdot\sin\alpha
\end{aligned} \tag{9}$$

After the change of coordinates, the motion equation of the robot has the following form:

$$\begin{aligned}
\dot{x} &= v\cdot\cos\vartheta \\
\dot{y} &= v\cdot\sin\vartheta \\
\dot{\varphi} &= \omega \\
\dot{v} &= a\cdot\cos\alpha = |\mathbf{a}_t| \\
\dot{\vartheta} &= \frac{1}{v}\cdot a\cdot\sin\alpha = \frac{1}{v}|\mathbf{a}_c| \\
\dot{\omega} &= \beta
\end{aligned} \tag{10}$$

It can be seen that the motion equation has become nonlinear. However, this form has a straightforward representation of the effect of tangential and centripetal accelerations $\mathbf{a}_t$ and $\mathbf{a}_c$ (see Figure 1) which determine the change in velocity magnitude and direction, respectively. Moreover, this form is better suited to our application because the set of allowed controls – the dynamic window – is defined in polar coordinates, as it will be described later in Section 5.

3.2 Configuration space

We recall some considerations regarding the configuration space (also referred to as *C-space*) of a planar robot with polygonal shape from [1]. The configuration space $\mathcal{C}$ of a 2-dimensional robot that can translate and rotate in the workspace $\mathcal{W} = \mathbb{R}^2$ is the manifold $\mathbb{R}^2 \times \mathbb{S}^1$. The obstacle region $\mathcal{O} \subset \mathcal{W}$ and the robot $\mathcal{R} \subset \mathcal{W}$ are given by a polygonal model (an example is depicted in Figure 2). The configuration space obstacle region $\mathcal{C}_{obs} \subset \mathcal{C}$ is defined as

$$\mathcal{C}_{obs} = \{\mathbf{q} \in \mathcal{C} \mid \mathcal{R}(\mathbf{q}) \cap \mathcal{O} \neq \emptyset\} \tag{11}$$

which consists of all configurations $\mathbf{q}$ for which the transformed robot $\mathcal{R}(\mathbf{q})$ is in collision with the obstacle region $\mathcal{O}$.

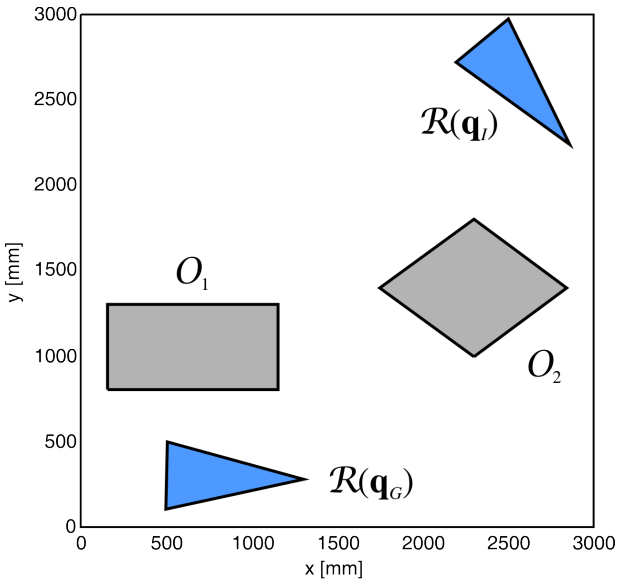

Fig. 2. Polygonal robot and obstacles. $\mathcal{O} = \mathcal{O}_1 \cup \mathcal{O}_2$ is the obstacle region, $\mathcal{R}(\mathbf{q}_I)$ is the robot in the initial configuration, $\mathcal{R}(\mathbf{q}_G)$ is the robot in the goal configuration.

An algorithm for collision detection of convex polygonal shapes is described in [1, pp. 164-166]. In the case of nonconvex obstacle and robot shapes these can be considered as the union of their convex parts. The nonconvex collision detection problem can be transformed into the collision detection of each convex robot part with each convex obstacle part. For a given configuration $\mathbf{q}$ the collision check has a complexity of

$$O\left(\sum_{i=1}^{n}\sum_{j=1}^{m} k_{R,i}\cdot k_{O,j}\right), \tag{12}$$

where n and m stand for the number of convex robot and obstacle subpolygons, respectively, $k_{R,i}$ denotes the vertex number of the i-th robot subpolygon ($i \in \{1,\ldots,n\}$) and $k_{O,j}$ stands for the vertex number of the j-th obstacle subpolygon ($j \in \{1,\ldots,m\}$). A useful method for minimal convex decomposition of simple polygons is proposed in [9].

To obtain an explicit model of C_{obs}, we discretize the configuration space with resolutions k_x, k_y and k_φ, which are positive integers. Let

$$\begin{aligned}\Delta\mathbf{q}_1 &= [x_{max}/k_x, \quad 0, \quad 0]^T \\ \Delta\mathbf{q}_2 &= [0, \quad y_{max}/k_y, \quad 0]^T \\ \Delta\mathbf{q}_3 &= [0, \quad 0, \quad \pi/k_\varphi]^T\end{aligned} \tag{13}$$

and let a grid point $\mathbf{q}'$ be expressed as

$$\mathbf{q}'(i,j,k) = i\Delta\mathbf{q}_1 + j\Delta\mathbf{q}_2 + k\Delta\mathbf{q}_3, \tag{14}$$

where $i \in \{0,\ldots,k_x\}$, $j \in \{0,\ldots,k_y\}$ and $k \in \{-k_\varphi,\ldots,k_\varphi\}$. Every grid point $\mathbf{q}'(i,j,k)$ is tested for collision and the C-space obstacle region is redefined as

$$C_{obs} = \{\mathbf{q} \in C \mid \mathcal{R}(f(\mathbf{q})) \cap O \neq \emptyset\}, \tag{15}$$

where $f(\mathbf{q})$ is a function that returns the grid point $\mathbf{q}'(i,j,k)$ that lies closest to $\mathbf{q}$:

$$\begin{aligned}i &= \lfloor x \cdot k_x/x_{max} + 0.5 \rfloor, \\ j &= \left\lfloor y \cdot k_y/y_{max} + 0.5 \right\rfloor, \\ k &= \left\lfloor \varphi \cdot k_\varphi/\pi + 0.5 \right\rfloor.\end{aligned} \tag{16}$$

Figure 3 illustrates C_{obs} for the environment depicted in Figure 2. Using this representation, the motion planning problem of a planar polygonal robot can be expressed as a planning problem of a single translating point in the (3-dimensional) configuration space. Note that the orientation angles $-\pi$ and $+\pi$ are identified, in other words the φ-axis "wraps around", which needs to be taken into account in further steps of the algorithm. It can be seen that the C-space obstacle region looks like as if it were built up of small "bricks" which is the result of discretization. The whole process has a complexity of

$$O\left(k_x \cdot k_y \cdot k_\varphi \cdot \sum_{i=1}^{n}\sum_{j=1}^{m} k_{R,i} \cdot k_{O,j}\right), \tag{17}$$

which means that at higher resolutions and in the case of complicated environment or robot shape the process of obtaining an explicit model for the configuration space is quite time consuming.

4 Navigation Function

As mentioned above, the task is to find a collision-free path in the C-space between initial and target configurations. The convergence can only be guaranteed if global information about free and occupied areas are taken into account. Similar to [5] and [7], we utilize a navigation function to achieve this. A navigation function (NF) is a real-valued function defined on the unoccupied part of the configuration space $C_{free} = C \setminus C_{obs}$, which has exactly one minimum, namely at the goal configuration $\mathbf{q}_G$.

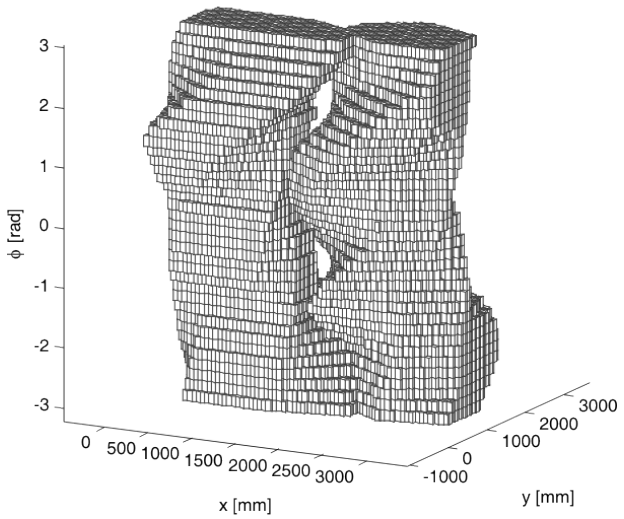

Fig. 3. Representation of C_{obs} using configuration space discretization ($k_x = k_y = 60$; $k_\varphi = 18$)

To define such a function, it is convenient to use the discretized configuration space model (14). Starting from $\mathbf{q}'_G = f(\mathbf{q}_G)$, the NF values at every reachable $\mathbf{q}' \in C_{free}$ are obtained by a wavefront propagation algorithm [1] (see Algorithm 1).

Algorithm 1 Wavefront propagation algorithm

1: Initialize $W_0 = \{\mathbf{q}'_G\}$; $s = 0$
2: Initialize $W_{s+1} = \emptyset$
3: **for all** $\mathbf{q}' \in W_s$ **do**
4: NF($\mathbf{q}'$) = s
5: Insert all unexplored neighbors of $\mathbf{q}'$ into W_{s+1}
6: **end for**
7: **if** $W_{s+1} = \emptyset$ **then**
8: **return**
9: **else**
10: $s := s + 1$
11: Go to step 2.
12: **end if**

We use *1-neighborhood* for wavefront propagation, which is defined as

$$N_1(\mathbf{q}') = \{\mathbf{q}' \pm \Delta\mathbf{q}_m \mid 1 \leq m \leq 3\}, \tag{18}$$

if $\mathbf{q}'$ is not a boundary grid point. Attention has to be paid to the boundaries during the wavefront propagation, especially if $\varphi \in \{-\pi, +\pi\}$ because these angles are identified, hence φ has actually no boundary. Note that we suppose that C_{free} is either simply connected or $\mathbf{q}_I$ and $\mathbf{q}_G$ are in the same connected submanifold of C_{free}. This is necessary in order to let the initial configuration to be reached by the wavefront propagation algorithm.

A cross section of the discrete navigation function at $\varphi = \varphi_G$ is shown in Figure 4. The different colors represent different navigation function values, the black area shows occupied configurations and the "×" mark stands for the goal configuration, where the wavefront propagation was started.

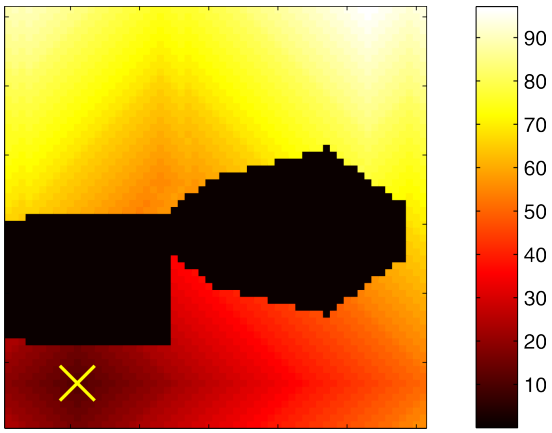

Fig. 4. Cross-section of NF($\mathbf{q}'$) at $\varphi = \varphi_G$

At this point, NF values have been assigned to grid points. As any configuration $\mathbf{q} \in C_{free}$ is allowed for the robot, NF values in all these points should be determined. In the simplest case, a zero-order interpolation could be applied. This would, however, be unsuitable for our purpose because the resulting function would be piecewise constant thus there could be motions that do not lower the function value while approaching the target. Instead, we use trilinear interpolation between grid points. It has the convenient property of having no local minima if neighboring points do not have equal values, which is ensured by the wavefront propagation algorithm together with the used 1-neighborhood.

5 Model Predictive Control Algorithm

The obstacle avoidance problem is considered as a constrained optimization problem over the control space of the robot, similar to (1). The objective function which has to be minimized is the navigation function itself, constructed as described in Section 4.

A control $\mathbf{u}(\cdot) : [t, t+T] \to [0, a_{max}] \times \mathbb{S}^1 \times [-\beta_{max}, \beta_{max}]$ has to be determined at every time instant t which minimizes the NF value at the end of a time horizon $[t, t+T]$.

$$\mathbf{u}(\cdot) = \arg\min_{\mathbf{u}(\cdot)} \mathrm{NF}\left(\mathbf{q}(t+T)\right) \tag{19}$$

where $\mathbf{q}(t+T)$ can be derived from the motion equation. At t the control $\mathbf{u}(t)$ has to be applied to the robot. Dynamic constraints of the robot are taken into account by velocity and acceleration bounds (5). Safety is ensured by a constraint of admissible controls. A control $\mathbf{u}(\cdot)$ is admissible, if

$$\begin{aligned} &\mathbf{q}(\tau) \notin C_{obs}, \\ &\forall \tau \in [t, t+T] \cup [t+T, t+T+T_{brake}] \end{aligned} \tag{20}$$

We assume that a maximal braking control $\mathbf{u}^*(\cdot)$ is applied in the time interval $[t+T, t+T+T_{brake}]$ where T_{brake} is the braking time needed to halt the robot beyond the horizon. The value of T_{brake} varies depending on the actual velocity at $t+T$. This means that only those controls are admissible that do not cause a collision inside the horizon and allow the robot to stop safely beyond the horizon.

To make the problem more tractable and computationally efficient, we assume discrete time and piecewise constant control. The horizon length is $T = hT_s$, $h \in \mathbb{Z}^+$, where T_s denotes sampling time. Piecewise constant control means that $\mathbf{u}(\tau) = \mathbf{u}(t)$ for all $\tau \in [t, t+T_s]$. Under this assumption we use a discrete-time version of the robot motion model (10):

$$\begin{aligned} x(t+T_s) &= x(t) + v(t) \cdot \cos\vartheta(t) \cdot T_s \\ y(t+T_s) &= y(t) + v(t) \cdot \sin\vartheta(t) \cdot T_s \\ \varphi(t+T_s) &= \varphi(t) + \omega(t) \cdot T_s \\ v(t+T_s) &= v(t) + a \cdot \cos\alpha \cdot T_s \\ \vartheta(t+T_s) &= \vartheta(t) + \frac{1}{v(t)} \cdot a \cdot \sin\alpha \cdot T_s \\ \omega(t+T_s) &= \omega(t) + \beta \cdot T_s \end{aligned} \tag{21}$$

Note that this motion model has a singularity at zero velocity thus when $v(t)$ is small (e.g. at the beginning or at the end of motion), (21) becomes numerical unstable. This inconvenience is eliminated by the following modification. If $v(t) \le \left|\frac{a \cdot \sin\alpha \cdot T_s}{2\pi}\right|$, then $\vartheta(t+T_s)$ is obtained by

$$\vartheta(t+T_s) = \vartheta(t). \tag{22}$$

In other words, if the change in ϑ would be greater than 2π, then ϑ is left unchanged.

During the optimization process we have to choose a control value at every discrete time instant that satisfies (20) and (5) and minimizes (19). To do this, the control space is also quantized, which results in a countable set of control vectors. The admissible control vectors that are inside the dynamic window are called *candidate controls*. We take each candidate control value from the dynamic window and calculate its effect to the robot for the duration of hT_s using (21). The control resulting in the smallest predicted navigation function value $\widetilde{\mathrm{NF}}\left(\mathbf{q}(t+hT_s)\right)$ will be chosen and applied to the robot in the time interval $[t, t+T_s]$.

The method has been tested in simulations. The results presented here were obtained for the robot already shown in Figure 2. The dynamic properties were chosen to $v_{max} = 0.75\mathrm{m/s}$, $\omega_{max} = 240\mathrm{deg/s}$, $a_{max} = 0.5\mathrm{m/s^2}$ and $\beta_{max} = 240\mathrm{deg/s}$. We use $T_s = 0.1\mathrm{s}$ sampling period and a variable horizon length

$$h(t) = \max\left(1 + \left\lceil \frac{|v(t)|}{a_{max} T_s} \right\rceil,\ 1 + \left\lceil \frac{|\omega(t)|}{\beta_{max} T_s} \right\rceil,\ 2\right). \tag{23}$$

Note that this results in a horizon length depending on the actual velocity which causes the robot to "look further" when traveling at higher velocities.

Figure 5 shows the resulting trajectory in the C-space from $\mathbf{q_I}$ to $\mathbf{q_G}$. As it can be seen, a smooth path is obtained due to the effect of limited accelerations.

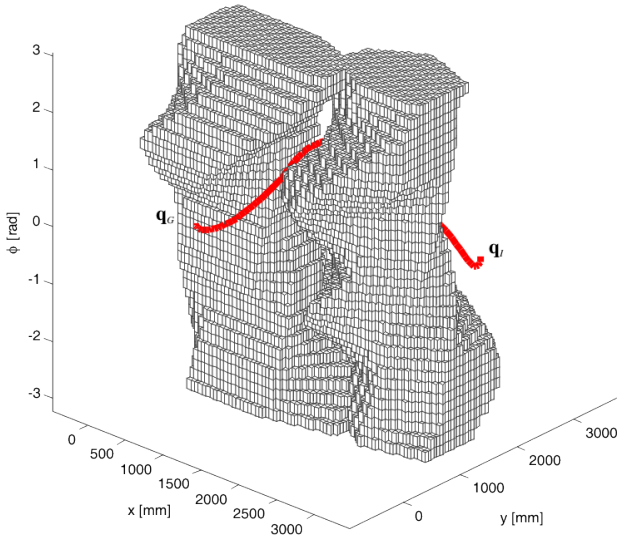

Fig. 5. Trajectory in the configuration space

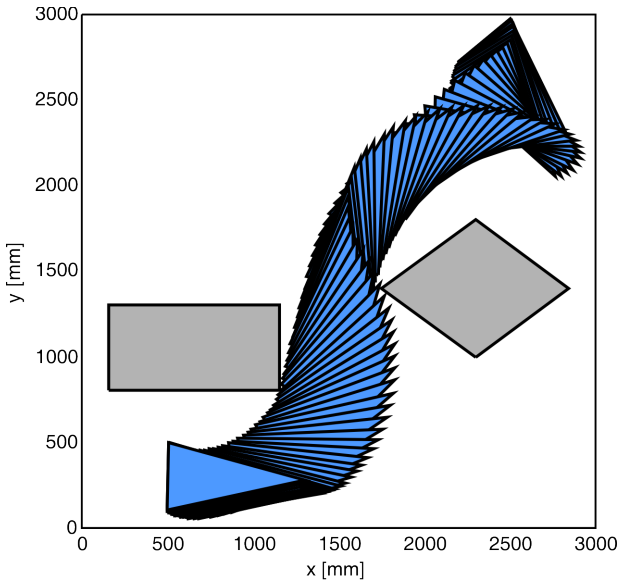

Fig. 6. Robot motion in the workspace

The same path transformed back into the workspace is depicted in Figure 6. Notice how the robot changes its orientation in order to be able to travel through the narrowing between the two obstacles.

In Figure 7, the translation velocity profile of the path can be observed. The robot approaches its maximum speed when traveling far from obstacles and keeps going fast even in the narrowing. The horizon length profile $h(t)$ is shown in the next diagram. As it can be seen, the horizon length follows the velocity according to (23). The evolution of NF values during motion is depicted in Figure 7 as well. The dotted line shows the predicted values $\widetilde{\mathrm{NF}}\,(\mathbf{q}(t + h(t)T_s))$, while the solid line illustrates $\mathrm{NF}\,(\mathbf{q}(t))$, the navigation function values actually realized by the robot. It can be seen that – as expected – the predicted values are always smaller than the actual ones. As a matter of fact, this is what makes the robot keep going until the goal is reached. It is interesting to examine the braking process while approaching

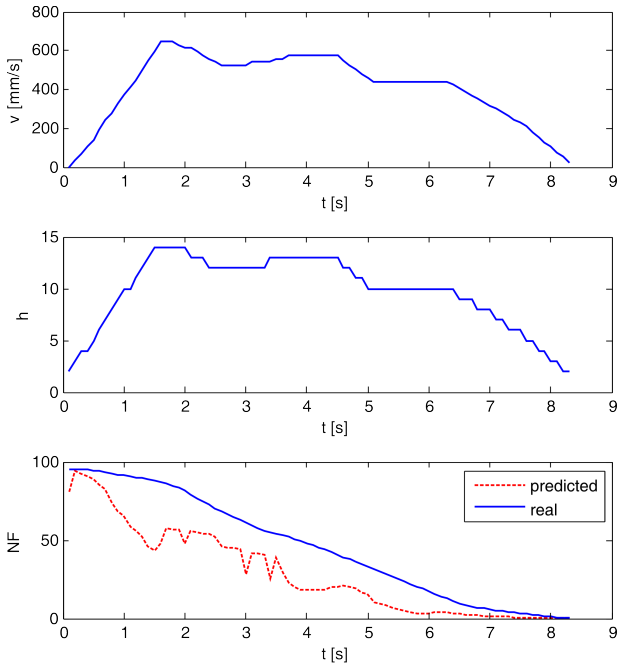

Fig. 7. Velocity profile and NF descent during motion

the goal (from $t = 6.3s$ to end). It can be seen that the predicted NF value does not change significantly, it is near zero during this phase. That is because the goal is already near enough to be "visible" inside of the horizon.

6 Conclusions

A dynamic window-based navigation approach was presented for holonomic polygonal robots. An iterative search is performed in the control space to obtain a control sequence that drives the robot to the goal safely and quickly. The search utilizes the idea of model predictive control, and a navigation function serves as optimization objective, defined on the configuration space. Although the search is performed locally in a short time horizon, the global distance information represented by the NF ensures convergence to the goal.

The only shortcoming of the method is the large computational cost of the C-space obstacle modeling process. This limits application in changing environments. One direction for future work is simplifying this process or evolving an analytical model for the C-space obstacle region.

Another improvement of this method would be the extension to robot models having differential constraints. Many real robot platforms are not holonomic but are equipped with e.g. differential drive or Ackerman steering. The method presented here cannot be applied directly to these because the navigation function that drives the robot towards the goal presumes that motion in every direction is allowed, which is only true in the absence of differential constraints.

References

1 **LaValle SM**, *Planning Algorithms*, Cambridge University Press; Cambridge, U.K., 2006. Available at http://planning.cs.uiuc.edu/.

2 **Khatib O**, *Real-time Obstacle Avoidance for Robot Manipulator and Mobile Robots*, The International Journal of Robotics Research, **5**, (1986), 90-98, DOI 10.1109/ROBOT.1985.1087247.

3 **Borenstein J, Koren Y**, *Real-time Obstacle Avoidance for Fast Mobile Robots*, IEEE Trans. Syst., Man, Cybern., **19**, (1989), 1179-1187, DOI 10.1109/21.44033.

4 **Borenstein J, Koren Y**, *The Vector Field Histogram - Fast Obstacle Avoidance for Mobile Robots*, IEEE Trans. Robot. Autom., **7**, (1991), 278-288, DOI 10.1109/70.88137.

5 **Fox D, Burgard W, Thrun S**, *The Dynamic Window Approach to Collision Avoidance*, IEEE Robot. Autom. Mag., **4**, (1997), 23-33, DOI 10.1109/100.580977.

6 **Schegel C**, *Fast Local Obstacle Avoidance under Kinematic and Dynamic Constraints for a Mobile Robot*, Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, (1998), 594-599, DOI 10.1109/IROS.1998.724683.

7 **Brock O, Khatib O**, *High-speed Navigation using the Global Dynamic Window Approach*, Proceedings of the IEEE International Conference on Robotics and Automation, (1999), 341-346, DOI 10.1109/ROBOT.1999.770002.

8 **Arras KO, Persson J, Tomatis N, Siegwart R**, *Real-Time Obstacle Avoidance For Polygonal Robots With A Reduced Dynamic Window*, Proceedings of the IEEE International Conference on Robotics and Automation, (2002), 3050-3055, DOI 10.1109/ROBOT.2002.1013695.

9 **Keil M, Snoeyink J**, *On the Time Bound for Convex Decomposition of Simple Polygons*, International Journal of Computational Geometry and Applications, **12**, (2002), 181-192.

10 **Ogren P, Leonard NE**, *A Convergent Dynamic Window Approach to Obstacle Avoidance*, IEEE Trans. Robot., **21**, (2005), 188-195, DOI 10.1109/TRO.2004.838008.

11 **Durrant-Whyte. H., Bailey T**, *Simultaneous Localisation and Mapping (SLAM): Part I The Essential Algorithms*, IEEE Robot. Autom. Mag., **13/2**, (2006), 99-110, DOI 10.1109/MRA.2006.1638022.

12 **Durrant-Whyte H, Bailey T**, *Simultaneous Localisation and Mapping (SLAM): Part II State of the Art*, IEEE Robot. Autom. Mag., **13/3**, (2006), 108-117, DOI 10.1109/MRA.2006.1678144.

13 **Berti H, Sappa AD, Agamennoni OE**, *Improved Dynamic Window Approach by Using Lyapunov Stability Criteria*, Latin American Applied Research, **38**, (2008), 289-298.

14 **Minguez J, Montano L**, *Extending Collision Avoidance Methods to Consider the Vehicle Shape, Kinematics, and Dynamics of a Mobile Robot*, IEEE Trans. Robot., **25**, (2009), 367-381, DOI 10.1109/TRO.2009.2011526.

15 **Koren Y, Borenstein J**, *Potential Field Methods and Their Inherent Limitations for Mobile Robot Navigation*, In: Proceedings of the IEEE International Conference on Robotics and Automation, 1991, pp. 1398-1404, DOI 10.1109/ROBOT.1991.131810.

16 **Simmons R**, *The Curvature-Velocity Method for Local Obstacle Avoidance*, In: Proceedings of the IEEE International Conference on Robotics and Automation, 1996, pp. 3375-3382, DOI 10.1109/ROBOT.1996.511023.

17 **Kiss D**, *Efficient Calculation of Navigation Functions for Obstacle Avoidance*, Proceedings of the Automation and Applied Computer Science Workshop 2011, (AACS'11), (June, 2011), 128-139.

18 **Kiss D, Tevesz G**, *A Receding Horizon Control Approach to Obstacle Avoidance*, Proceedings of the 6th IEEE International Symposium on Applied Computational Intelligence and Informatics (SACI 2011), (May, 2011), 397-402.